Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion

Walter Daelemans
Antal van den Bosch

ABSTRACT We describe an approach to grapheme-to-phoneme conversion which is both *language-independent* and *data-oriented*. Given a set of examples (spelling words with their associated phonetic representation) in a language, a grapheme-to-phoneme conversion system is automatically produced for that language which takes as its input the spelling of words, and produces as its output the phonetic transcription according to the rules implicit in the training data. We describe the design of the system, and compare its performance to knowledge-based and alternative data-oriented approaches.

1 Introduction

Grapheme-to-phoneme conversion is an essential module in any text-to-speech system. It can be described as a function mapping the spelling form of words to a string of phonetic symbols representing the pronunciation of the word. The largest part of research on this process focuses on developing systems that implement various levels of language-specific linguistic knowledge (especially morphological and phonotactic knowledge, but often also syntactic knowledge). It is generally assumed that this is essential to solving the task. MITalk (Allen et al., 1987) is a classic example of such a knowledge-based approach for English; for Dutch, Morpa-cum-Morphon (Heemskerk & van Heuven, 1993, Nunn & van Heuven, 1993) can be considered state-of-the-art. A clearly disadvantageous consequence of the knowledge-based strategy is the fact that it requires a large amount of handcrafting of linguistic rules and data during development. Furthermore, language-specificity of a grapheme-to-phoneme model tends to be incompatible with reusability of the developed implementation, i.e., for each language, a specific set of rules and principles has to be found in order to successfully run the model.

In this article we describe an implemented grapheme-to-phoneme con-

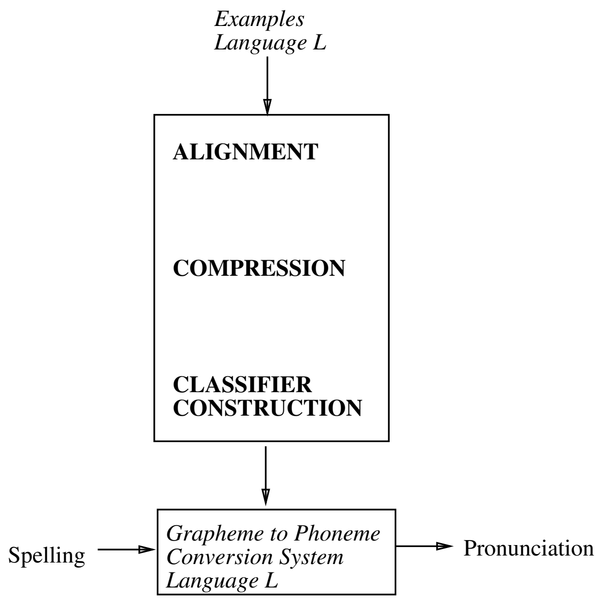

FIGURE 1. Overview of the architecture.

version architecture and explore to what extent it allows data-oriented induction of a grapheme-to-phoneme mapping on the basis of examples, thereby alleviating the expensive linguistic engineering phase. Input to our system is a set of spelling words with their associated pronunciations in a target phonemic or phonetic alphabet (the training data). Spelling and pronunciation do not have to be aligned. The phonetic transcription can be taken from machine-readable or scanned dictionaries, or from automatic phoneme recognition. The words may represent text in context (when effects transgressing word boundaries have to be modeled) or isolated words. Output of the system is a grapheme-to-phoneme conversion system which takes as its input the spelling of words, and produces as its output the phonetic or phonemic transcription according to the rules implicit in the training data (see Figure 1 for an overview of the architecture).

The approach has a number of desirable properties:

1. It is *data-oriented*. The output system is constructed automatically from the training data, thereby effectively removing some of the knowledge acquisition bottlenecks. Knowledge-Based solutions to the problem need considerable handcrafting of phonological and morphological data structures, analysis and synthesis programs.

2. It is in principle *language-independent and reusable*. Versions of the

system for French, Dutch and English have been constructed automatically using the same architecture on different sets of training data. In linguistic approaches, the handcrafting has to be redone for each new language. Languages with other writing systems may exist, however, for which the approach is less well suited.

3. It achieves a *high accuracy*. Output of the Dutch version has been extensively compared to the results of a state-of-the-art 'hand-crafted' linguistic system. The data-oriented solution proved to be significantly more accurate in predicting phonetic transcriptions of previously unseen words.

2 Design of the System

The system consists of the following modules:

1. *Automatic alignment*: strings of spelling symbols and strings of phonetic symbols have to be made of equal length in order to be processed by the other modules.
2. *Automatic training set compression*: part of the training data is represented in a compact way using tree structures and information gain.
3. *Automatic classifier construction*: using the compacted training data, a classifier is constructed that extrapolates from its memory structures to new, unseen input spelling strings.

We will discuss these stages in turn. Compression and classifier construction are achieved at the same time in our current implementation, and will be discussed together.

2.1 Alignment

The spelling and the phonetic trancription of a word often differ in length. The grapheme-to-phoneme conversion module described in the next section demands, however, that the two representations be of equal length, so that each individual graphemic symbol can be mapped to a single phonetic symbol. The problem is to align the two representations in such a way that graphemes or strings of graphemes are consistently associated with the same phonetic symbols. This is not a trivial task: if this alignment has to be done by hand, it is extremely labour-intensive. Consider the example alignments of the words 'rusty' and 'rookie':

graphemes	r	u	s	t	y	r	oo	k	ie
	|	|	|	|	|	|	|	|	|
phonemes	r	ʌ	s	t	ɪ	r	u	k	i

Whereas the alignment of 'rusty' is very straightforward (i.e., five one-to-one grapheme-to-phoneme mappings), the alignment of 'rookie' involves the knowledge that it the 'oo' cluster maps to the /u/-phoneme, and that the 'ie' cluster maps to the /i/-phoneme. No other partitioning of the spelling string is allowed, or at least intuitively correct. Our automatic alignment algorithm attempts to make the length of a word's spelling string equal to the length of its transcription by adding *null* phonemes to the transcriptions. Nulls have to be inserted in the transcription at those points in the word where a grapheme cluster maps to one phoneme. In the example of the word 'rookie', this would be done as follows ('–' depict phonemic nulls):

graphemes	r	o	o	k	i	e
	|	|	|	|	|	|
phonemes	r	u	–	k	i	–

Note that it is arbitrary whether one uses the /ru-ki-/ aligment or the /r-uk-i/ aligment (i.e., whether one chooses to map the last or the first of a grapheme cluster to a phonetic null), as long as it is done consistently. Alignments such as /ruki - -/ or /- - ruki/ should never be generated, because they imply highly improbable mappings. An alignment should always follow the principles (i) that its grapheme-to-phoneme mappings are viable (i.e., that they can be motivated intuitively or linguistically); (ii) that the combination of all mappings within the alignment has a maximal probability; and (ii) that it is consistent with aligments of other, similar words.

The first part of the algorithm automatically captures the probabilities of all possible phoneme-grapheme mappings in an association matrix. For each letter in the spelling string, the association with the phoneme that occurs at the same position in the *unaligned* transcription is increased; furthermore, if a spelling string is longer than its transcription, phonemes which precede the letter position are also counted, as they may be associated with the target letter as well. In other words, the algorithm determines for each letter in the graphemic string, all phonemes in the transcription to which it *may* map. Furthermore, these phonemes are weighted differently. As shown in the diagram below, the algorithm shifts the unaligned phonemic word from the left aligned position to the right aligned position. Taking the 'k' as an example, the algorithm adds a score of 8 to the association between 'k' and /i/, a score of 4 to the association between 'k' and /k/, and a score of 2 to the association between 'k' and /u/. Note that shifting is repeated at most 3 times; at the third shift, which may occur with longer words, associations are increased by a score of 1. Other values for these weights result in slightly (but not significantly) worse results. The idea behind this weighting is the simple assumption that left-aligned phonemic transcriptions *generally* contain more correct grapheme-to-phoneme mappings than right-aligned transcriptions.

graphemes	r	o	o	k	i	e	association
	|	|	|	|	|	|	score
phonemes	r	u	k	i	_	_	8
phonemes, 1 right shift	_	r	u	k	i	_	4
phonemes, 2 right shifts	_	_	r	u	k	i	2

Although a lot of noise is added to the association matrix by including associations that are less probable (e.g., in our example, the mapping between 'k' and /i/), the use of this shifting association 'window' ensures that the most probable associated phoneme is always captured in this window. More importantly, it turns out that the 'intuitive' or 'linguistically appropriate' mappings receive the highest scores in the end. When all words of the data base are processed this way, the association scores in the association matrix are converted into association probabilities.

The second part of the alignment algorithm generates for each pair of unaligned spelling and phoneme strings all possible (combinations of) insertions of null phonemes in the transcription. For each hypothesised string, a total association probability is computed by multiplying the scores of all individual letter-phoneme association probabilities of each hypothesised alignment. The hypothesis with the highest total association probability is then taken as output of the algorithm.

The resulting alignment, albeit very consistent, is not always identical to the intuitive alignment applied by human coders. To test its efficacy, we compared classification accuracy of the complete system when using a hand-aligned training set as opposed to the automatically aligned training set. The results indicate that there is no significant difference in classification accuracy: the alignments result in equally accurate systems. The resulting IG-Tree (see following section) is on average about 3% larger with the automatically generated alignment, however.

2.2 IG-Trees: Compression and Classifier Construction

Rules and Patterns

The way grapheme-to-phoneme conversion works in our system can be seen as optimised, generalised lexical lookup. Reasoning is based on *analogy* in the overall correspondence of the writing system and the pronunciation of a certain language. Words that are spelled similarly, are pronounced similarly. The system automatically learns to find those parts of words on which similarity matching can safely be performed. This is perhaps best illustrated in the example of the word <behave>. An 'analogical' model which operates with a certain similarity metric, and which has already encountered (and learned) roughly similar words such as <shave>, <beehive>, and <have>, and perhaps even <behave> itself, will certainly have a number of clues as to how <behave> is pronounced. However, in this example, problems may arise with the <a> of <behave>. If the similarity matcher of the analogical model decides to retrieve the pronunciation of the word <have> as the pronunciation of <-have> in <behave>, the incorrect pronunciation /bihæv/ would result. Our system does not take such overgeneralisation risks. The model is extremely sensitive to context, in the sense that it will have stored the knowledge that <have> is not enough context to be certain of the pronunciation of the <a>. Instead, the system will look for more contextual information. In a current implementation of the system trained on English words, the system decides to take /e^j/ as output only when it finds the sub-word chunk <ehave> in the input word. Note that this system has encountered the word <behave> during training, but that for the case of the pronunciation of the <a>, it was not necessary to store the complete word, as there were no other words with the sub-word chunk <ehave> with a different pronunciation of the <a>. In sum, the system stores single letter – phoneme correspondences with a minimal context that is sufficient to be certain that the mapping is unambiguous (in the training material).

Each of these sub-word – phoneme correspondences can be seen as a context-sensitive rewrite rule, which rewrites a letter in context to a phoneme. As the context may be of any width, many of these rewrite rules are much more specific than a typical rule in a rule-based grapheme-to-phoneme module would be; many even contain whole words. Although one would be tempted to categorise such an approach as rule-based, it could equally well be regarded as a lexical approach. Some of the rules are so specific that it would make more sense to call them lexical patterns. The rules are a compressed version of the text-to-speech corpus it is trained on. After training, they contain in a compressed format complete knowledge of the pronunciation of all words of the learning material (lossless compression), except for homographs such as <read> (pronounced as /rɛd/ or /rid/), of which only one pronunciation is stored. This is certainly a shortcoming for languages such as Russian, where pronunciation depends on stress placement, which in its turn depends on lexical properties which cannot be deduced from the form of the word (which is the only information the current implementation of our system uses). We will return to this problem in our conclusion. See Yarowsky (this volume) for an approach to homograph disambiguation using decision lists, which could be integrated with our approach.

To illustrate the appearance of automatically extracted rules, Table 1 lists some examples that the model extracted from English data and French data.

Table 1: Examples of automatically extracted sub-word – phoneme correspondences, with their associated phonemes, from English and French data. Example words containing the rules are given. Dots represent unused context positions; underscores represent word boundaries.

English											Phoneme	Example Word
.	.	.	.	.	v	o	.	.	.	.	v	**v**oucher
.	.	.	e	s	i	d	e	n	.	.	ə	pres**i**dent
.	.	.	.	w	o	_	.	.	.	.	u	tw**o**
.	.	.	_	h	a	v	e	_	.	.	æ	h**a**ve
French											Phoneme	Example Word
.	.	.	.	.	ç	.	.	.	.	.	s	fran**ç**ais
.	.	_	_	b	e	a	u	x	.	.	o	b**e**aux
.	.	.	.	v	i	n	_	.	.	.	E	v**i**n
.	.	.	.	n	c	o	.	.	.	.	k	fran**c**ophone

From Table 1, it can clearly be seen that some correspondences express very general pronunciation knowledge, whereas others are used to disambiguate between only a few words, e.g., <esiden> – /ə / discriminates <president> from <reside>. There is no clear distinction between rules and lexical patterns, they are regarded as extremes in a continuum.

Compressing knowledge into an IG-Tree

The system does not actually store a large list of context-sensitive rewrite rules and lexical patterns. It compresses the information contained in these rules even more by storing them in a decision tree. Each rule is represented as a path in this tree. A path consists of a starting node which represents the target letter that is to be mapped to a phoneme; the consecutive nodes represent the consecutive context letters. The order in which these letters are attached to the path is governed by computing their overall relative importance in disambiguating the mapping. This is done using Information Gain, a computational metric based on Information Theory (hence the name IG-Tree). A description of this metric is given in Daelemans *et al.* (1994). Computation of the Information Gain of context positions renders a result that is constant for all corpora used. Trivially, the focus letter itself is the most important 'context' letter. The further the context position is removed from the focus letter, the less important that position is

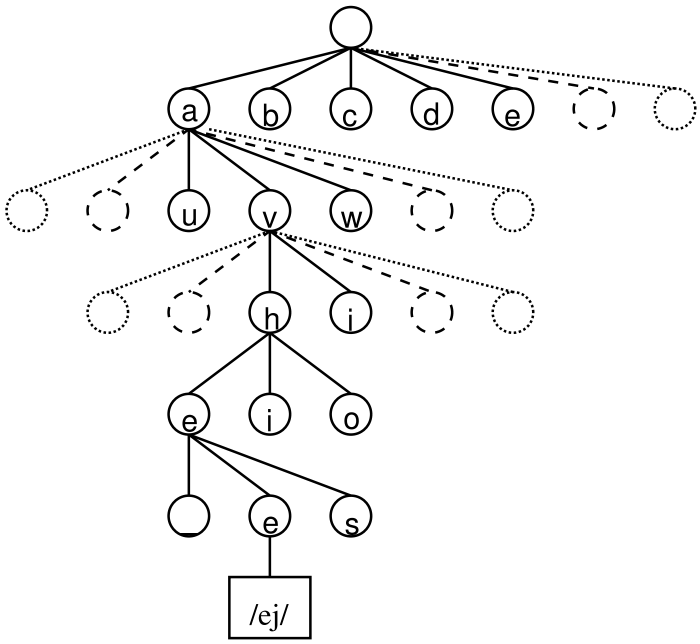

FIGURE 2. Retrieval of the pronunciation of < a >, /e^{j}/, of the word < *behave* >. The path represents the minimally disambiguating context < *ehave* >.

for disambiguation, on the average. Furthermore, there is an as yet unexplained difference between right and left context: right context positions are computed to be slightly more important than their respective left context positions. In practice, this leads to an ordering in which the first character on the right is the first context expansion, i.e., the first node down the tree. Then follows the first character on the left, then the second character on the right, then the second character on the left, and this alternating pattern simply repeats. To visualise the way in which knowledge is organised in the decision tree, Figure 2 displays the part of the tree in which the pronunciation of the <a> in the word <behave> is stored.

With the <a>-node as starting point, the node labelled with the first character on the right, <v>, is the second node accessed in the path. Then, the <h>-node, the first character to the left of the <a>, is taken. At that point, the only possible extensions stored in the tree are <have>, <havi> (from <having>) and <havo> (from <havoc>); the pronunciation at that point is still ambiguous. Then, the <e>-node is accessed, which leaves open the extensions <_have> (underscores depict word boundaries), <ehave>, and <shave>. As mentioned earlier, at the next step, the model, retrieves the unambiguous phonemic mapping /e^{j}/, when the final <e> node is reached.

It can be seen that the depth of a path reflects in a certain sense the ambiguity of the mapping it represents. End nodes near the top of the decision tree typically belong to highly regular pronunciations. For example, the French model contains at the top layer of the tree the end node <ç>, as this special character always maps to /s/, regardless of the context. An example of an extremely ambiguous mapping is that of the first <o> of <photograph>, /o/, which has the competitor word <photography>, in which the <o> maps to /ə/. In this case, a context to the right of width 8 is needed for disambiguation.

Best Guess Strategy

In our approach, all spelling-to-phonology knowledge contained within the learning material is stored lossless, with the exception of homographs, of which only one pronunciation is kept. The rule-based aspect of the decision tree, however, enables the model also to generalise to new cases. To retrieve the pronunciation of a word that was not in the learning material, each letter of the new word is taken as a starting point of tree search. The search then traverses the tree, up to the point where the search successfully meets an end node, or where the search fails as the specific context of the new word was not encountered in the learning material, and consequently was not stored as a path in the tree. In the first case, the phonemic label of the end node is simply taken as the phonemic mapping of the new word's letter. In the second case, the exact matching strategy is taken over by a *best guess* strategy.

In present implementations of the system, the best guess strategy is implemented in a straightforward way. When building a path in the tree, the construction algorithm constantly has to check whether an unambiguous phonemic mapping has been reached. At each node, the algorithm searches in the learning material for all phonemic mappings of the path at that point of extension. In cases when there is more than one possible phonemic mapping, the algorithm computes what is the most *probable* mapping at that point. Computation is based on occurrences: the most frequent mapping in the learning material is preferred (in case of ties, a random choice is made). This extra information is stored with each non-ending node. When a search fails, the system returns the most probable phonemic mapping stored in the node at which the search fails.

3 Related Approaches

The information gain metric used to select context features while building the IG-Tree is used in a similar way in C4.5 decision tree learning (Quinlan, 1993). The main difference with C4.5's approach to decision tree learning is the fact that our model computes the ordering only once for the complete tree, whereas in C4.5 the ordering is computed at every node. Another difference is that in the IG-tree, nodes are created until all training set ambiguity is resolved (there is no pruning of the tree).

In earlier versions of this system (Daelemans & van den Bosch, 1993, Van den Bosch & Daelemans, 1993), a different approach was taken to handle strings which were not found in the IG-Tree. A similar effect to defaults at leaf nodes was achieved by combining the compression and IG-Tree building with a form of similarity-based reasoning (based on the k-nearest neighbour decision rule, see e.g. Devijver & Kittler, 1982). During training, a memory base is incrementally built consisting of *exemplars*. In our domain of grapheme-to-phoneme conversion, an exemplar consists of a string of graphemes (one focus grapheme surrounded by context graphemes) with one or more associated phonemes and their distributions (as there may exist more phonemic mappings for one graphemic string). During testing, a test pattern (a graphemic string) is matched against all stored exemplars. If the test pattern is in memory, the category with the highest frequency associated with it is used as output. If it is not in memory, all stored exemplars are sorted according to the similarity of their graphemic string pattern to the test pattern. The (most frequent) phonemic mapping of the highest ranking exemplar is then predicted as the category of the test pattern. Daelemans & Van den Bosch (1992) extended the basic nearest-neighbour algorithm by introducing Information Gain as a means to assigning different weights to different grapheme positions when computing the similarity between training and test patterns (instead of using a distance metric based on overlap of patterns).

In the combination of IG-Tree and the Information Gain-aided nearest neighbour algorithm, the IG-Tree classification algorithm stops when a graphemic string is not found in the tree. Instead of using the information on the most probable phoneme at the non-ending leaf node, the nearest-neighbour algorithm takes over and searches in its exemplar base for the best matching exemplar to the graphemic pattern under consideration. The phoneme associated with the best matching exemplar is then taken as 'best guess'.

We experimented both with the nearest neighbour approach independent from the IG-Tree algorithm, and with the combination of the two. We found that the current approach (select most probable category at ambiguous leaf nodes), being computationally simpler and conceptually more elegant, achieves the same generalisation accuracy. We therefore adopted this simpler approach. The current approach is also as accurate as using the nearest neighbour technique independently.

Earlier work on the application of nearest neighbour approaches (Memory-Based Reasoning, Stanfill & Waltz, 1986; Stanfill, 1987) to the phonemisation problem using the NetTalk data (MBRTalk), showed a better performance than NetTalk (Sejnowski & Rosenberg, 1987) itself, however at the cost of an expensive, domain-dependent computational measure of dissimilarity that seems to be computationally feasible only when working on a massive parallel computer like the Connection Machine. Another analogy-based system (or rather a hybrid combination of case-based reasoning and relaxation in a localist interactive activation network) is PRO (Lehnert, 1987). However, the reported performance of this system is not very convincing, neither is the need for a combination of connectionist and case-based techniques apparent. Dietterich and Bakiri (1991) systematically compared the performance of ID3 (a predecessor of C4.5, Quinlan 1993) and BP on the NetTalk data. Their conclusion is that BP consistently outperforms ID3 because the former captures statistical information that the latter does not. However, they demonstrate that ID3 can be extended to capture this statistical information. Dietterich and Bakiri suggest that there is still substantial room for improvement in learning methods for text-to-speech mapping, and it is indeed the case that our approach significantly outperforms BP.

The application of compression techniques such as our IG-Tree to the phonemisation problem has not yet been reported on as such in the literature. In Golding & Rosenbloom (1991), the interaction of rule-based reasoning and case-based reasoning in the task of pronouncing surnames is studied. It is claimed that a hybrid approach is preferable, in which the output of the rules is used *unless* a compelling analogy exists in the case-base. If a compelling analogy is found, it overrides the rule output. In this approach, the (hand-crafted) rules are interpreted as implementing the defaults, and the cases the *pockets of exceptions*. Our IG-Tree method works along a different dimension: both default mappings (rules) and pockets of exceptions are represented in the IG-Tree in a uniform way.

4 Evaluation

In this section, we compare the accuracy of our approach to the knowledge-based approach and to alternative data-oriented approaches for Dutch grapheme-to-phoneme conversion. More detailed information about the comparison can be found in and Van den Bosch & Daelemans (1993).

4.1 Connectionism

In our approach, explicit use is made of analogical reasoning. Back-propagation learning in feedforward connectionist networks (BP), too, uses similarity (or analogy), but more implicitly. An input pattern activates an output pattern which is similar to the activation pattern of those items that are similar to the new item. Complexity is added by the fact that an intermediate hidden layer of units “redefines” similarity by extracting features from the activation patterns of the input layer.

Automatic learning of grapheme-to-phoneme conversion of English (NETtalk, Sejnowski & Rosenberg, 1987) has been acclaimed as a success story for BP. The approach was replicated for Dutch in NetSpraak (Weijters & Hoppenbrouwers, 1990). It is therefore appropriate to compare our alternative data-oriented approach to BP.

The performance scores on randomly selected, unseen test words (generalisation accuracy) show a best score for the IG-Tree approach. Similar results were obtained for different training and test sets.

Model	Generalisation Accuracy on Phonemes
BP	91.3
IG-Tree	95.1

4.2 *The Linguistic Knowledge-Based Approach*

The traditional linguistic knowledge-based approach of grapheme-to-phoneme conversion has produced various examples of combined rule-based and lexicon-based models. The developers of all of these models shared the assumption that the presence of linguistic (phonotactic, morphological) knowledge is essential for a grapheme-to-phoneme model to perform at a reasonably high level.

In Morpa-cum-Morphon (Heemskerk & van Heuven, 1993, Nunn & van Heuven, 1993), a state-of-the-art system for Dutch, grapheme-to-phoneme conversion is done in two steps. First, Morpa decomposes a word into a list of morphemes. These morphemes are looked up in a lexicon. Each morpheme is associated with its category and a phonemic transcription. The phonemic transcriptions of the consecutive morphemes are concatenated to form an underlying phonemic representation of the word. Morphon then applies a number of phonological rules to this underlying representation, deriving the surface pronunciation of the word. The system is the result of a five-year research effort, sponsored by the Dutch government and industry, and is generally acclaimed to be the best system available.

We applied the IG-Tree method to the same test data as they used to evaluate their system, in order to make a comparison possible. Again, we see that the IG-Tree scores significantly higher[1].

Model	Generalisation Accuracy on Words
IG-Tree	89.5
morpa-cum-morphon	85.3

[1] Note that meanwhile, the performance of Morpa-cum-Morphon on the benchmark was boosted to 88.7% by incorporating a data-oriented probabilistic morphological analysis component (Heemskerk, 1993).

5 Conclusion

The most surprising result of our research is that an extremely simple method (based on compressing a training set) yields the best accuracy results (judged by measuring *generalisation* accuracy), suggesting that previous knowledge-based approaches as well as more computationally expensive learning approaches to at least some aspects of the problem were overkill.

The system described constructs a grapheme-to-phoneme conversion module on the basis of an unaligned corpus of spelling words and their phonetic representations. The approach is data-oriented (eliminates linguistic engineering), language-independent (reusable for different dialects or languages), and accurate (when compared to knowledge-based and alternative data-oriented methods).

Current limitations are the absence of word stress computation (but see Daelemans et al. 1994 for a compatible data-oriented approach to this problem) and sentence accent computation (for which syntactic, semantic/pragmatic and discourse information is required). The present word pronunciation modules output by our system can be combined with existing approaches to this problem, however. This raises the problem of modularity in the design of data-oriented (learning) approaches to grapheme-to-phoneme conversion. Should stress assignment be integrated with grapheme-to-phoneme conversion or should two separate systems be trained to learn these two aspects of the problem, one using as input the output of the other? This is a subject for further research.

Finally, as we mentioned earlier, our approach fails miserably in all cases where different pronunciations correspond to the same spelling string (as soon as two pronunciations differ in their spelling, if only in a single letter, there is no problem, but in that case, useful generalisations may be missed). For languages such as Russian this is clearly an important shortcoming. However, our approach is not limited to using only spelling information. Additional features (e.g. lexical category) can be added to the spelling features as input, and these would then be used in the construction of the IG-tree without any change to the present system. In that case, the added features would be used in generalisation as well.

Acknowledgments: We thank the two anonymous referees for useful comments on an earlier version of this article.

6 References

[AHK87] J. Allen, S. Hunnicut, and D. H. Klatt. *From Text to Speech: The MITalk System.* Cambridge University press, Cambridge, U.K., 1987.

[BD93] A. van den Bosch & W. Daelemans, Data-oriented methods for grapheme-to-phoneme conversion. *Proceedings of the Sixth conference of the European chapter of the ACL*, ACL, 45-53, 1993.

[DB92] W. Daelemans & A. van den Bosch. Generalization performance of backpropagation learning on a syllabification task. In M. Drossaers & A. Nijholt (Eds.), *Proceedings of the 3rd Twente Workshop on Language Technology.* Enschede: Universiteit Twente, 27-37, 1992.

[DB93] W. Daelemans & A. van den Bosch. TABTALK: Reusability in Data-oriented grapheme-to-phoneme conversion. *Proceedings of Eurospeech*, Berlin, 1459-1466, 1993.

[DGD94] W. Daelemans, S. Gillis, & G. Durieux. The Acquisition of Stress, a data-oriented approach. *Computational Linguistics* 20 (3), 421-451, 1994.

[DK82] P.A. Devijver, & J. Kittler. *Pattern recognition. A statistical approach.* London: Prentice-Hall, 1982

[DB91] T. G. Dietterich & G. Bakiri. Error-correcting output codes: a general method for improving multiclass inductive learning programs. *Proceedings AAAI-91*, Menlo Park, CA, 572-577, 1991.

[GR91] A. R. Golding & P. S. Rosenbloom. Improving rule-based systems through Case-Based Reasoning. *Proceedings AAAI-91*, Menlo Park, CA, 22-27, 1991.

[HH] J. Heemskerk & V. J. van Heuven. MORPA, a lexicon-based MORphological PArser. In V.J. van Heuven and L.C.W. Pols (Eds.), *Analysis and synthesis of speech; strategic research towards high-quality text-to-speech generation.* Berlin: Mouton de Gruyter, 1993.

[Hee93] J. Heemskerk. A probabilistic context-free grammar for disambiguation in morphological parsing. In *Proceedings EACL-93*, Utrecht, 1993.

[Leh87] W. Lehnert. Case-based problem solving with a large knowledge base of learned cases. In *Proceedings AAAI-87*, Seattle, WA, 1987.

[NH] A. Nunn & V. J. van Heuven. MORPHON, lexicon-based text-to-phoneme conversion and phonological rules. In V.J. van Heuven and L.C.W. Pols (Eds.), *Analysis and synthesis of speech; strategic research towards high-quality text-to-speech generation.* Berlin: Mouton de Gruyter, 1993.

[Qui93] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1993.

[SR87] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce English text. *Complex Systems*, 1, 145-168, 1987.

[SW86] C. W. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:12, 1213-1228, 1986.

[Sta87] C. W. Stanfill. Memory-based reasoning applied to English pronunciation. *Proceedings AAAI-87*, Seattle, WA, 577-581, 1987.

[WH90] A. Weijters & G. Hoppenbrouwers. NetSpraak: een neuraal netwerk voor grafeem-foneem-omzetting. *Tabu*, 20:1, 1-25, 1990.